Global Optimization based on Contractor Programming

Jordan Ninin, Gilles Chabert

Proceedings of MAGO 2014, pp. 1 – 4.

Global Optimization based on Contractor Programming

Jordan Ninin[1] and Gilles Chabert[2]

[1] *ENSTA-Bretagne, LabSTIC, IHSEV team, 2 rue Francois Verny, 29806 Brest, France* jordan.ninin@ensta-bretagne.fr

[2] *Ecole des Mines de Nantes, LINA, TASC team, 4 rue Alfred Kastler, 44300 Nantes, France* gilles.chabert@mines-nantes.fr

Abstract In this paper, we will present a general pattern based on contractor programming for designing a global optimization solver. This approach allows to solve problems with a wide variety of constraints. The complexity and the performance of the algorithm rely on the construction of contractors which characterize the feasible region.

Keywords: Global Optimization, Interval Arithmetic, Contractor Programming

1. Introduction

Considering sets in place of single points is not a common point of view in the Mathematical Programming communities. In contrast, this interpretation is the key concept in global optimization based on interval arithmetic and in constraint programming.

Faced to complexity and diversity of real-life problems, it is hard to find a general pattern or unique algorithm for solving all of them. Some algorithms deal with disjunctive constraints, others with dynamic constraints, others with uncertainties. Furthermore, when real-life problems merge different kinds of constraints, we often need to remove a part of the model just because our solvers do not accept it. Unfortunately, these cases are not rare.

In this talk, we will present a general pattern based on contractor programming that allows to handle a wide variety of problems. In Section 2, we recall the definitions related to interval arithmetic and contractor programming. Then, we present a short subset of some standard contractors and how we can combine and merge them. In Section 4, a global optimization algorithm based on contractors is detailed.

2. Definitions

Since the book of Moore in 1966 [9], many techniques have been developed based on interval arithmetic. More generally, all these techniques can be considered as Set-Membership Methods. These methods and algorithms do not consider single numerical values, or floating-point numbers, but manipulate sets. The interval arithmetic offers a solid theoretical basis to represent and to calculate with subsets of $\mathbb{R}^n$.

An interval is a closed connected subset of $\mathbb{R}$. A non-empty interval $[\mathbf{x}]$ can be represented by its endpoints: $[\mathbf{x}] = [\underline{x}, \overline{x}] = \{x : \underline{x} \leq x \leq \overline{x}\}$ where $\underline{x} \in \mathbb{R} \cup \{-\infty\}$, $\overline{x} \in \mathbb{R} \cup \{+\infty\}$ and $\underline{x} \leq \overline{x}$. The set of intervals will be denoted by $\mathbb{IR}$, and the set of n-dimensional interval vectors, also called *boxes*, will be denoted by $\mathbb{IR}^n$.

Definition 1. *Let $\mathbb{X} \subseteq \mathbb{R}^n$ be a feasible region.*
The operator $\mathcal{C}_{\mathbb{X}} : \mathbb{IR}^n \to \mathbb{IR}^n$ is a contractor for $\mathbb{X}$ if:

$$\forall [\boldsymbol{x}] \in \mathbb{IR}^n, \begin{cases} \mathcal{C}_{\mathbb{X}}([\boldsymbol{x}]) \subseteq [\boldsymbol{x}], & \textit{(contraction)} \\ \mathcal{C}_{\mathbb{X}}([\boldsymbol{x}]) \cap \mathbb{X} \supseteq [\boldsymbol{x}] \cap \mathbb{X}. & \textit{(completeness)} \end{cases}$$

The concept of contractor is very broad for integrating and interfacing mixed techniques [6]. Contractors are directly inspired form constraint programming. A contractor is defined for a feasible region $\mathbb{X}$ and its purpose is to eliminate a part of a domain which is not in $\mathbb{X}$.

Proposition 2. *The operator $\mathcal{C} : \mathbb{IR}^n \to \mathbb{IR}^n$ is a contractor for the equation $f(x) = 0$, if:*

$$\forall [\boldsymbol{x}] \in \mathbb{IR}^n, \begin{cases} \mathcal{C}([\boldsymbol{x}]) \subseteq [\boldsymbol{x}], \\ \forall x \in [\boldsymbol{x}],\ f(x) = 0 \Longrightarrow x \in \mathcal{C}([\boldsymbol{x}]). \end{cases}$$

The basic implementation of a contractor for a numerical constraint is the forward-backward algorithm, also called constraint propagation technique or FBBT or HC4-Revise [4, 8, 11]. This algorithm is the basic block of contractor programming.

All set operators can be extended to contractors. For example, the intersection of two contractors creates a contractor for the intersection of these two sets. In the same way, the hull of two contractors creates a contractor for the disjunction of these constraints.

Definition 3. *Let $\mathbb{X}$ and $\mathbb{Y} \subseteq \mathbb{R}^n$ be two feasible regions.*

$$\begin{array}{rl} \textit{Intersection:} & (\mathcal{C}_{\mathbb{X}} \cap \mathcal{C}_{\mathbb{Y}})([\boldsymbol{x}]) = \mathcal{C}_{\mathbb{X}}([\boldsymbol{x}]) \cap \mathcal{C}_{\mathbb{Y}}([\boldsymbol{x}]) \\ \textit{Union:} & (\mathcal{C}_{\mathbb{X}} \cup \mathcal{C}_{\mathbb{Y}})([\boldsymbol{x}]) = \mathcal{C}_{\mathbb{X}}([\boldsymbol{x}]) \cup \mathcal{C}_{\mathbb{Y}}([\boldsymbol{x}]) \\ \textit{Composition:} & (\mathcal{C}_{\mathbb{X}} \circ \mathcal{C}_{\mathbb{Y}})([\boldsymbol{x}]) = \mathcal{C}_{\mathbb{X}}(\mathcal{C}_{\mathbb{Y}}([\boldsymbol{x}])) \\ \textit{Fixed Point:} & \mathcal{C}^{\infty} = \mathcal{C} \circ \mathcal{C} \circ \mathcal{C} \circ \dots \end{array}$$

Many known techniques can be cast into contractors. All linear relaxation techniques can be considered as contractors. Such a contractor is constructed by intersecting the input box with the polyhedral hull created by the linear relaxation. This intersection is obtained by solving at most $2n$ linear programs, see [2, 10] for details.

3. Non-conventional Contractors

The main interest of contractors is the ability to deal with constraints that are difficult to combine or to formulate mathematically. For example, if the variable corresponds to a position on a map, it is very simple to make the intersection of a given box with an area of a map, while it would have been cumbersome to describe this area by a mathematical equation.

Another common case is the possibility of corrupted data. If a set of constraints are based on physical data, it is not uncommon that some of this data are wrong. For example, only 80 % of constraints are acceptable, without knowing which ones. In this situation, the q-relaxed intersection of contractors can be applied to this problem:

Definition 4. *The q-relaxed intersection of m subsets $\mathbb{X}_1, \dots, \mathbb{X}_m$ of $\mathbb{R}^n$ is the set of all $x \in \mathbb{R}^n$ which belong to at least $(m - q)$ $\mathbb{X}_i$. We denote it by $\mathbb{X}^{\{q\}} = \bigcap^{\{q\}} \mathbb{X}_i$.*

Since the q-relaxed intersection is a set operator, we can extend this notion to contractors: $\left(\bigcap^{\{q\}} \mathcal{C}_{\mathbb{X}_i}\right)([\mathbf{x}]) = \bigcap^{\{q\}} (\mathcal{C}_{\mathbb{X}_i}([\mathbf{x}]))$. This contractor allows to model the possibility of invalid constraints: it can also be used for robust optimization.

In [5], Carbonnel et al. found an algorithm with a complexity $\theta(nm^2)$ to compute a box which contains the q-relaxed intersection of m boxes of $\mathbb{R}^n$. This algorithm can be interpreted as an implementation of the q-relaxed intersection of m contractors.

Another possibility is to project a subset of $\mathbb{R}^n$ over one or more dimensions. For example, if a constraint needs to be satisfied for all values of a parameter in a given set, such as $\{x \in \mathbb{R}^n \ : \ \forall t \in \mathbb{X} \subseteq \mathbb{R}^m, g(x,t) \leq 0\}$, few solvers are available to deal with it. Another example is when a constraint needs to be satisfied for at least one value of the parameter, such as $\{x \in \mathbb{R}^n \ : \ \exists t \in \mathbb{X} \subseteq \mathbb{R}^m, g(x,t) \leq 0\}$.

Two operators are defined on contractors. The first one is the *projection-intersection* and the second one is the *projection-union*.

Definition 5. *Let $\mathbb{X} \subseteq \mathbb{R}^n$, $\mathbb{Y} \subseteq \mathbb{R}^m$, $\mathbb{Z} \subseteq \mathbb{R}^p$, with $\mathbb{Z} = \mathbb{X} \times \mathbb{Y}$. Let $\mathcal{C}$ be a contractor for the set $\mathbb{Z}$. We define $\mathcal{C}^{\cap\mathbb{Y}}$ the* Projection Intersection *of $\mathbb{Z}$ over $\mathbb{X}$ and $\mathcal{C}^{\cup\mathbb{Y}}$ its* Projection Union *by:*

$$\forall x \in \mathbb{R}^n, \ \left\{ \begin{array}{l} \mathcal{C}^{\cap\mathbb{Y}}([\boldsymbol{x}]) = \bigcap_{y\in\mathbb{Y}} \pi_{\mathbb{X}} \left(\mathcal{C}([\boldsymbol{x}] \times \{y\})\right), \\ \mathcal{C}^{\cup\mathbb{Y}}([\boldsymbol{x}]) = \bigcup_{y\in\mathbb{Y}} \pi_{\mathbb{X}} \left(\mathcal{C}([\boldsymbol{x}] \times \{y\})\right). \end{array} \right.$$

with $\pi_{\mathbb{X}}$ the projection of $\mathbb{Z}$ over $\mathbb{X}$.

Proposition 6. *Let $\mathcal{C}$ be a contractor for a set $\mathbb{Z}$. $\mathcal{C}^{\cap\mathbb{Y}}$ is a contractor for the set $\mathbb{X} = \{x \ : \ \forall y \in \mathbb{Y}, (x,y) \in \mathbb{Z}\}$ and $\mathcal{C}^{\cup\mathbb{Y}}$ is a contractor for the set $\mathbb{X} = \{x \ : \ \exists y \in \mathbb{Y}, (x,y) \in \mathbb{Z}\}$.*

The Projection-Intersection contractor contracts each part of $[\mathbf{x}]$ which are contracted by $\mathcal{C}([\mathbf{x}] \times \{y\})$ for any $y \in \mathbb{Y}$. Indeed, each part $[\mathbf{a}]$ of $[\mathbf{x}]$, such as $\exists y \in \mathbb{Y}, \ ([\mathbf{a}], y) \notin \mathbb{Z}$, can be removed. Thus, each part $[\mathbf{b}]$ of $[\mathbf{x}]$, such as $\forall y \in \mathbb{Y}, \ ([\mathbf{b}], y) \in \mathbb{Z}$, is kept. A similar argument proves Proposition 6 for the Projection-Union contractor.

4. Global Optimization Algorithm

The implementations of all the previous contractors are available in our library IBEX [1, 3]. Thus, given a physical problem, the user can construct a contractor for the feasible region $\mathbb{X}$ of his problem. We denote this contractor $\mathcal{C}_{out}$. Moreover, using the counterparts of set-membership operators for contractors (cf. Definition 3), we can construct in the same way a contractor for the negation of $\mathbb{X}$. This contractor is denoted by $\mathcal{C}_{out}$. The only required mathematical expression is the objective function, f_{cost}.

Given a box $[\mathbf{x}] \in \mathbb{R}^n$, $\mathcal{C}_{out}([\mathbf{x}])$ removes from $[\mathbf{x}]$ a part that does not contain a feasible solution. In the same way, $\mathcal{C}_{in}([\mathbf{x}])$ removes from $[\mathbf{x}]$ parts which are entire feasible; i.e. $([\mathbf{x}]/\mathcal{C}_{in}([\mathbf{x}])) \subseteq \mathbb{X}$. Thus, $([\mathbf{x}]/\mathcal{C}_{in}([\mathbf{x}]))$ is a feasible subset and we can perform a global optimization without constraint on it. If this step succeeds, this set can be discarded: indeed, if a new best current solution is found, we save it and it is proved that this set does not contain a better solution; else it is directly proved that no better solution can be found in this set $([\mathbf{x}]/\mathcal{C}_{in}([\mathbf{x}]))$.

The following algorithm describes a simple implementation pattern for a global optimization solver based on contractors. This algorithm is inspired from the *SIVIA* Algorithm (Set-Inversion Via Interval Analysis), which is used to compute the feasible set in a domain [7].

The inputs are an initial domain $[\mathbf{x}] \in \mathbb{IR}^n$, $\mathcal{C}_{out}$ a contractor for $\mathbb{X}$, $\mathcal{C}_{in}$ a contractor for $\overline{\mathbb{X}}$ and f_{cost} an objective function. The outputs are $\tilde{f}$, the global minimum value found and $\tilde{x}$, a global minimum. A boolean variable b is added for each element of $\mathcal{L}$ to indicate if the element is included in the feasible region.

$(\tilde{x}, \tilde{f})$ = ***OptiCtc*** $([\mathbf{x}], \mathcal{C}_{out}, \mathcal{C}_{in}, f_{cost})$:
  $\tilde{f} := +\infty$, denotes the current upper bound for the global minimum;
  $\mathcal{L} := \{([\mathbf{x}], false)\}$, initialization of the data structure of the stored elements;
  Let $\mathcal{C}_f$ a contractor based on the constraint $\{x \ : \ f_{cost}(x) \leq \tilde{f}\}$;
  Repeat until a stopping criterion is fulfilled:
    Extract from $\mathcal{L}$ an element $([\mathbf{y}], b)$,

Bisect the considered box $[\mathbf{y}]$: $[\mathbf{y}_1], [\mathbf{y}_2]$,
for $j = 1$ to 2 :
if (b = false) *then*
Contract $[\mathbf{y}_i]$ with $\mathcal{C}_{out} \cap \mathcal{C}_f$,
$[\mathbf{y}_{tmp}] := [\mathbf{y}_i]$,
Contract $[\mathbf{y}_i]$ with $\mathcal{C}_{in}$,
Add ($[\mathbf{y}_i]$,false) in $\mathcal{L}$.
$[\mathbf{y}_{tmp}] := [\mathbf{y}_{tmp}]/[\mathbf{y}_i]$,
else
Contract $[\mathbf{y}_i]$ with $\mathcal{C}_f$,
$[\mathbf{y}_{tmp}] := [\mathbf{y}_i]$,
Try to ***find*** the global optimum without constraint in $[\mathbf{y}_{tmp}]$,
if the search succeeds in a limited time *then*
Update $\tilde{f}$ and $\tilde{x}$.
else
Add $([\mathbf{y}_{tmp}], true)$ in $\mathcal{L}$.
end.

Further refinements can improve the behavior of the algorithm, but we must keep in mind that most of time, the performances depend of the problem itself. With this algorithm, the improvements which can have a real and direct impact are in the implementation of the contractors and in the relevance of the model.

5. Summary

This paper introduces the use of the contractors for designing a global optimization algorithm. These concepts will be illustrated to minimize the consumption of two robots following non-linear parametric trajectories and subject to two constraints: conflict avoidance and validation of at least 80% checkpoints.

References

[1] IBEX : *a C++ numerical library based on interval arithmetic and constraint programming*. http://www.emn.fr/z-info/ibex/.

[2] I. Araya, G. Trombettoni, and B. Neveu. *A contractor based on convex interval taylor*. In Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Springer, 2012.

[3] I. Araya, G Trombettoni, B. Neveu and G Chabert. *Upper Bounding in Inner Regions for Global Optimization under Inequality Constraints*. Journal of Global Optimization, 2014, to appear.

[4] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Waechter, *Branching and bounds tightening techniques for non-convex MINLP*, Optimization Methods & Software, vol. 24, pp. 597-634, 2009.

[5] C. Carbonnel, G. Trombettoni, P. Vismara and G. Chabert, *Q-intersection algorithms for robust parameter estimation*, in submission.

[6] G. Chabert and L. Jaulin, *Contractor programming*, Artificial Intelligence, vol. 173, n. 11, pp. 1079-1100, 2009.

[7] L. Jaulin and E. Walter. *Set inversion via interval analysis for nonlinear bounded-error estimation*. Automatica, vol. 29, n.4, pp. 1053-1064, 1993.

[8] F. Messine, *Deterministic global optimization using interval constraint propagation techniques*, RAIRO-Operations Research, vol. 38, pp. 277-293, 2004.

[9] R.E. Moore, Interval Analysis, Prentice-Hall Inc., Englewood Cliffs, 1966.

[10] J. Ninin, P. Hansen, and F. Messine, *A reliable affine relaxation method for global optimization*. Cahier du GERAD, in submission.

[11] X.-H. Vu, D. Sam-Haroud, and B. Faltings, *Enhancing numerical constraint propagation using multiple inclusion representations*, Annals of Mathematics and Artificial Intelligence, vol. 55, pp. 295-354, 2009.